AFRL-IF-RS-TR-2004-262
Final Technical Report
September 2004

TEMPLATE ENHANCEMENT THROUGH KNOWLEDGE ACQUISITION (TEMPLE)

University of Southern California at Marina Del Rey

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J735

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-262 has been reviewed and is approved for publication

APPROVED: /s/

JAMES F. REILLY
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	SEPTEMBER 2004	Final Apr 00 – Dec 03

4. TITLE AND SUBTITLE
TEMPLATE ENHANCEMENT THROUGH KNOWLEDGE ACQUISITION (TEMPLE)

5. FUNDING NUMBERS
C - F30602-00-2-0513
PE - 63760E
PR - ATEM
TA - P0
WU - 03

6. AUTHOR(S)
Yolanda Gil

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California at Marina Del Rey
Information Science Institute
4676 Admiralty Way
Marina Del Rey California 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-262

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: James F. Reilly/IFSA/(315) 330-3333/ James.Reilly@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

In this contract they developed Constable, a tool that enables users to extend existing templates by adding new constraints to be checked as they create plans from those templates. The constraints acquired specify desired qualities of the final plan, such as requiring the presence of a fires task for each attack task plan, or that the use of a resource or the overall cost be under a given threshold. Initially, the system has a pre-defined suite of constraints relevant to the domain that can be used to check a plan as it is being created by the user. The system also has an easy-to-use interface based on constrained English that guides users step by step (in a wizard-style interaction) to define additional new constraints that they would like the system to check.

14. SUBJECT TERMS
Active Templates, Constable, Intelligent Interfaces, Knowledge Acquisition, Planning Constraints

15. NUMBER OF PAGES
30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1 Project Overview

Military planners typically work under time pressure to develop complex plans that must take into account a myriad of constraints, ranging from rules of engagement to commander's guidance to feasibility of resources. It is not easy for users to keep in mind all the constraints and check that the current version of the plan satisfies all of them, especially as the plan evolves over time and when several people are involved in the process. These plans often end up ignoring important constraints, which may be only noticed much later and at times only as they are executed. Some of these problems may be quite severe and require further modifications to the plan. Tools that help users improve their plans by enforcing given constraints will be an important part of the kind of plan development environment that Active Templates aims to provide.

It is very important to provide users with the ability to add new constraints to these systems, since it is not possible to specify all constraints exhaustively before hand. Many of the constraints depend on the specifics of the operation, the commander's guidance, or their own past experience. For example, when a new rule of engagement is introduced, users need to enter that in the system as a new constraint that their plans need to satisfy. Acquiring this kind of knowledge from users had been an important goal of our previous research in knowledge acquisition within the EXPECT architecture for developing knowledge-based systems (http://www.isi.edu/ikcap/expect). We have already used EXPECT to create *plan constraint checking tools in several domains to help users improve the consistency and quality of manually created plans*, including air campaign planning and Army Course of Action (COA) critiquing. This work has allowed us to learn directly from military users what kinds of customizations and new constraints they would want to add to a system.

In this contract we developed **Constable**, a tool that enables users to extend existing templates by adding new constraints to be checked as they create plans from those templates. The constraints acquired specify desired qualities of the final plan, such as requiring the presence of a fires task for each attack task in the plan, or that the use of a resource or the overall cost be under a given threshold. Initially, the system has a pre-

defined suite of constraints relevant to the domain that can be used to check a plan as it is being created by the user. The system also has an easy-to-use interface based on constrained English that guides users step by step (in a wizard-style interaction) to define additional new constraints that they would like the system to check. Constable also allows users to specify possible corrective actions that may be appropriate when a constraint is violated. For example, if a user adds a new plan step that causes a violation on a constraint on the overall cost, the system could suggest to the user to reconsider adding that step or to modify another existing step to reduce its current cost.

Constable contributed to several important goals of the Active Templates program: 1) enable ordinary end users to *tailor default templates* by adding constraints caused by the specific requirements of the operation and their own preferences for what plans they believe will work better; 2) provide *incremental payoff* since the system immediately checks the added constraints, and will check as many constraints as the user takes the time to add; 3) an *active environment to create plans*, since the system is able to check the added constraints and be active in taking appropriate corrective actions**.** As a result, users can create plans *faster and of better quality*, since the system is thorough in ensuring that all the specified constraints are satisfied, detect right away when they are not, and suggest (or take) appropriate corrective actions.

This work leverages from three key technologies that we had investigated in EXPECT:

- **Guiding users to enter new knowledge step by step in following typical knowledge acquisition dialogues.** Entering new knowledge typically requires adding several individual but related pieces of information. In order to guide users to specify correctly all the necessary information, we will identify *knowledge acquisition scripts* that will capture typical sequences of steps that users follow in adding new constraints to a system, and use them to guide users through all the necessary steps in a wizard-style interaction.
- **Exploiting domain-independent background knowledge that captures general principles about well-formed plans and types of user preferences so that users do not have to formulate new constraints from scratch**. From our past experience and from analysis of the literature we have identified a set of general principles for

defining and evaluating planning constraints. TEMPLE uses explicit representations of these principles to guide a user adding a new plan constraint by framing it within these principles and to apply general-purpose code fragments to the new constraint.

- **Handling the interaction with users by integrating NL generation and direct manipulation techniques to create structured editors that rely on constrained English**. TEMPLE also includes an English-based editor that helps users specify the knowledge needed for checking and applying a constraint. The editor uses a restricted set of internal constructs that is mapped to a subset of English. The user is only shown paraphrases with the constrained-English representation, and is able to enter knowledge through direct manipulation of the English text. Users select only those fragments of text that correspond to meaningful expressions in the system's internal language. The editor creates a set of alternatives that the expression might be changed to and present them as fragments of text to the user.

We have used these techniques in our previous work in ISI's EXPECT knowledge acquisition framework. As part of our work on the DARPA HPKB program, we have developed an initial prototype of an acquisition tool for EXPECT that combines an English-based editor with knowledge acquisition scripts based on background knowledge about plan critiques.

We worked very closely with experts in Special Operations to develop demonstration scenarios appropriate for planners in this domain, as well as designing customizations of the interface and software that would be appropriate for Special Operations planning.

Our system was integrated with other software in the Active Templates program. For example, the synchronization matrix provided by SofTools can be used as an interface for users to invoke Constable to add a new temporal constraint, such as that any attack operations in the plan must start at least two hours after the beginning of the execution of the operation.

Our experience in military domains shows that many requirements and restraints that need to be taken into account during planning are specific to the operation, the commander's guidance, or the user's preferences. By enabling users to customize a system by adding these new constraints, users will be able to develop plans faster and of

higher quality, especially when they make hundreds of changes as they refine the plan over time and with severe time constraints.

2 Accomplishments

Our work concentrated on four major areas: 1) acquisition of planning constraints from users; 2) specification of a plan constraint language to be used by ourselves and others in the Active Templates program; 3) visualization as a means for elicitation of constraints from users; 4) integration of our software with SofTools and support for demonstrations in the domain of Special Operations planning.

Each of these topics is elaborated in the sections that follow and corresponding appendices.

2.1 Constable: Acquiring Planning Constraints from Users

Constable is a *Constraint Editing Tool* that helps users modify or add knowledge to an intelligent computer system. Constable is able to critique plans created in Softools. This is done with a number of customizable checks on the plan called *constraints*. Examples of constraints are that an asset used in a movement should be either a CRRC or a Mk-V, or that the lunar illumination should be less than 0.5.

This section provides an overview of how Constable works, a detailed user guide is provided in Appendix I.

Figure 1 shows an example of a plan that Constable is checking, if a constraint is violated the field is highlighted in red.

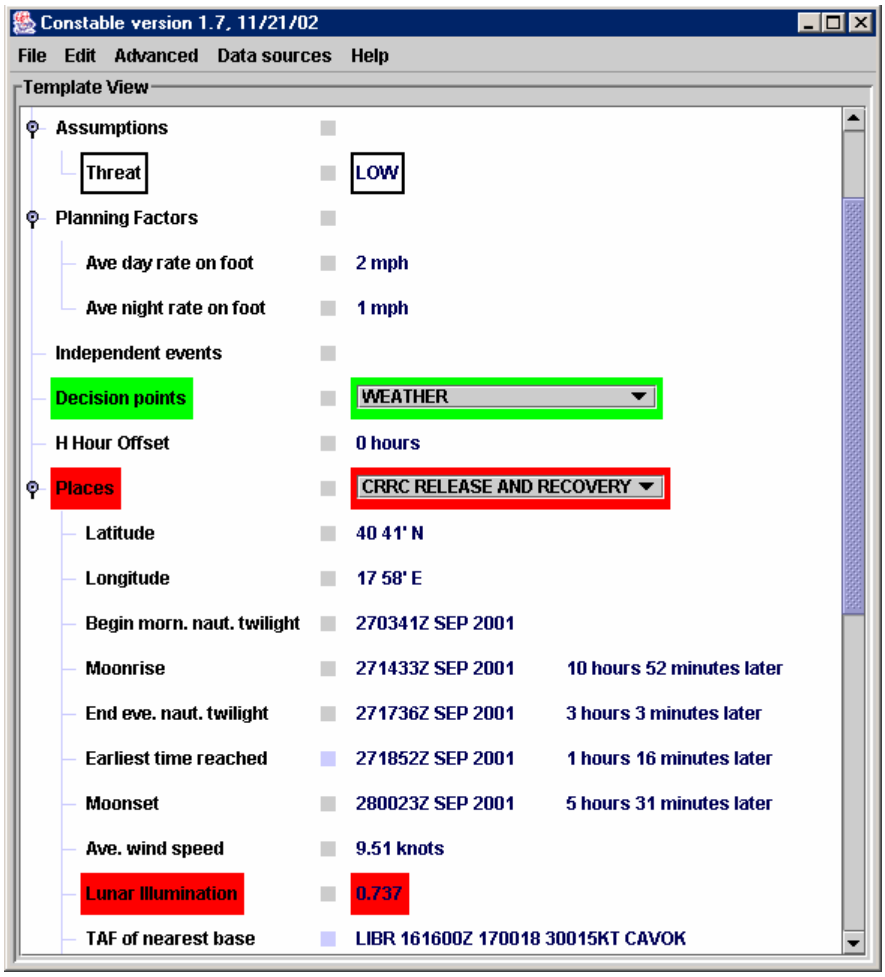

Figure 1. Viewing the current plan in Constable, highlighting constraint violations in red.

In general, Constable can use constraints that check a field in many different ways, making use of background information about locations and equipment, and other planning factors and assumptions. Constable gives help for defining certain kinds of constraints, for example a constraint that involves a numerical value and checks against a minimum or maximum value. If the value of a field is not numerical, for instance "CRRC", the constraint can check if it is a member of a set of preferred values, or a set of values to be avoided. A field and its constraints can be edited using several tools from the menu that appears when you right-click over the field, as described in the next section.

Units may have different preferences about the lunar illumination, depending on the activities or enemy capabilities. In this section and the next one, we increase the maximum value in the constraint that Constable checks for the illumination from 0.3 to 0.8.

Figure 2 shows all the information that Constable used to compute and check the lunar illumination field. When field values are computed and constraints are checked, Constable may combine information from many sources, including other constraints that may solve a part of the problem. The first panel shows that constraints from two general groups were used. The second panel, labeled 'Live Data Sources', shows that some of the information came from querying external sources such as web sites. The third panel, labeled 'Information used', summarizes the data that was used in order to compute and check the field. In this case, the illumination is for the place called 'AFSB-GOLD' and its latitude, longitude and start-time (a Softools field) were used. The fourth panel, labeled 'Assumptions' shows some of the constraints that were used to check the field, including a 'maximum allowed value' which will be changed in this example.

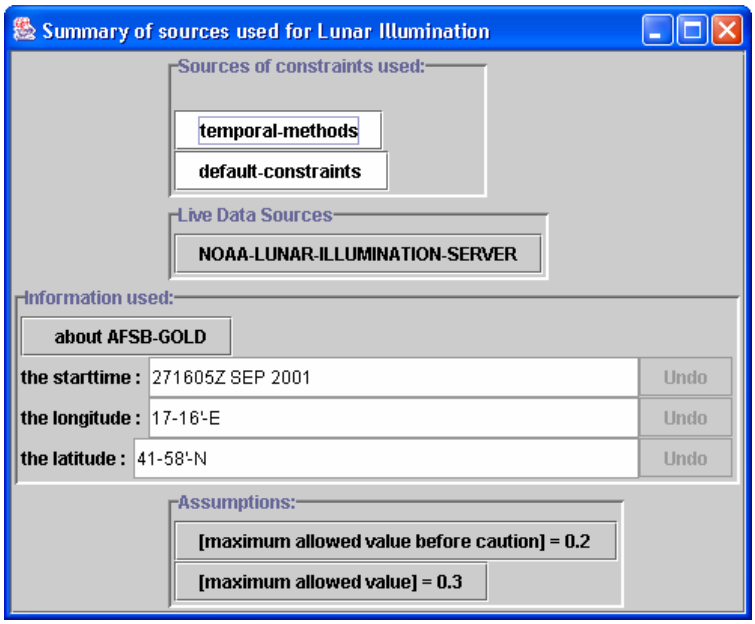

Figure 2. Looking up the sources of information used to evaluate a constraint on the plan in Constable.

Constable shows users the definitions of constraints as English paraphrases of the formal, internal definition of the constraint. Constable can be used to change the way that fields or constraints are computed, as well as to change constants as in the previous section. Suppose that, because of a range of mountains, moonrise is effectively an hour later than the value from astronomical calculations. Rather than give a fixed value, we need to add an hour to any value that is computed. First, the user would look at the constraint definition in Constable. Figure 3 shows a snapshot on how constraint information is organized. It would shows how the field value is computed for a location in a Softools plan: first it computes the earliest time the place is reached, then it makes a query to an information agent for the beginning of nautical twilight at this time and location.

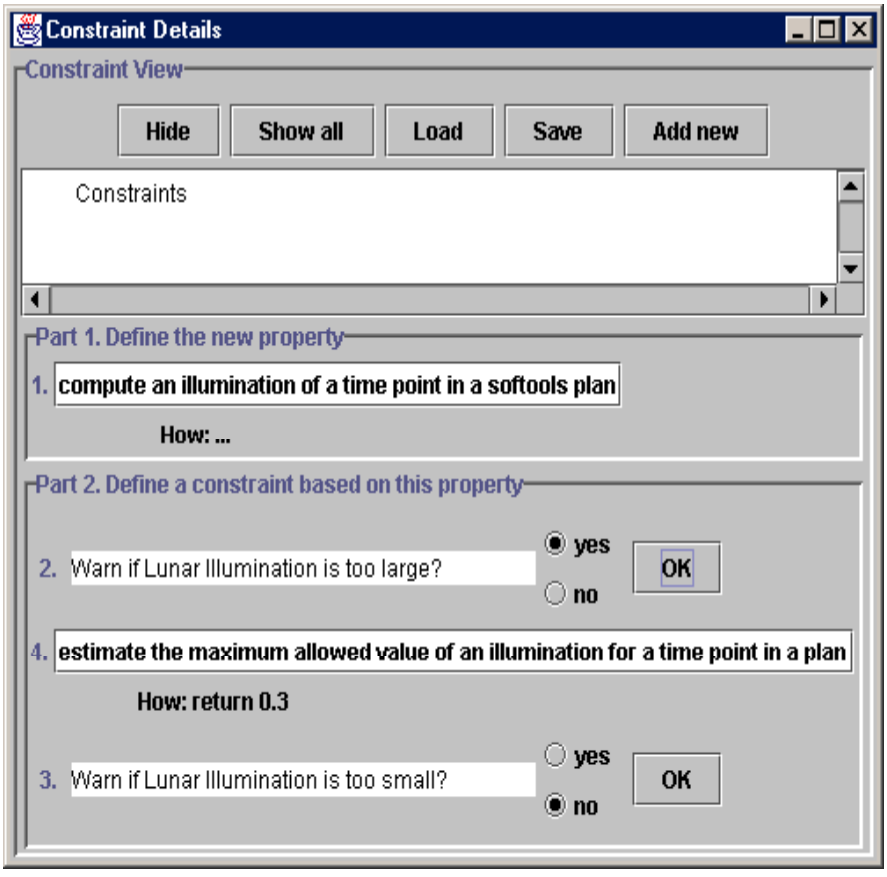

Figure 3. Viewing the details on how a constraint is evaluated in Constable.

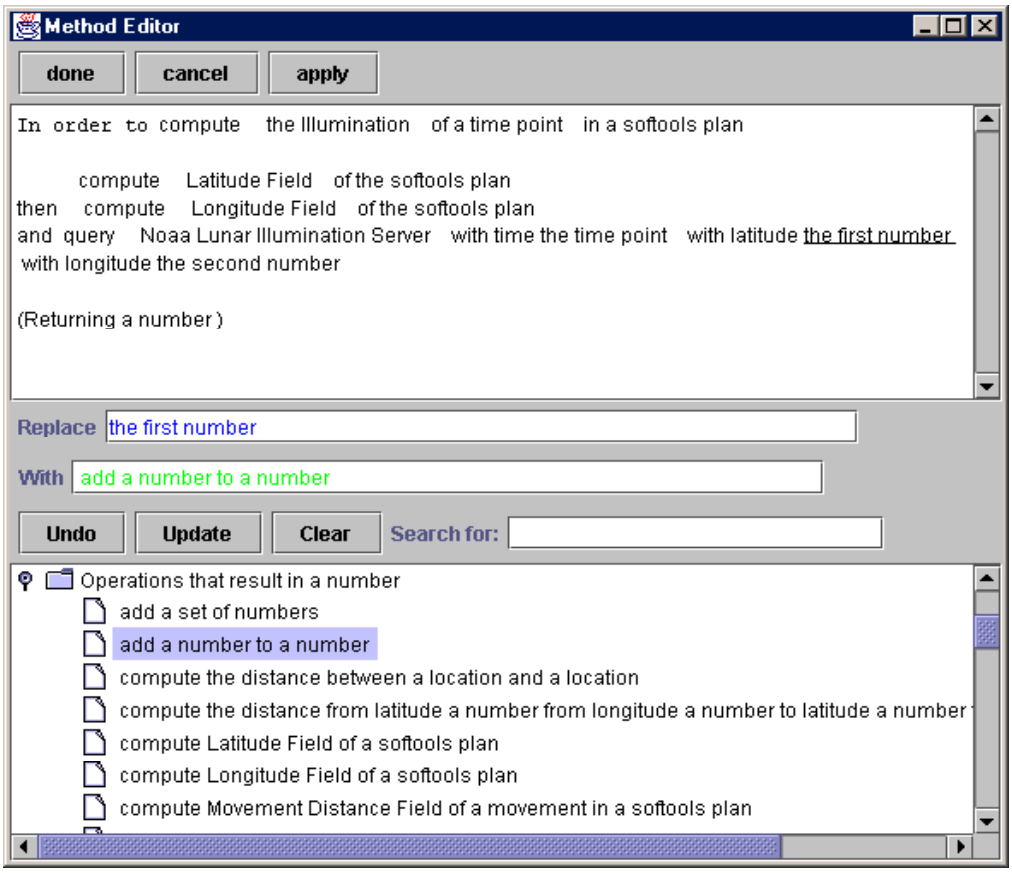

Figure 4. Modifying a constraint in Constable with a text editor.

Figure 4 shows a snapshot of the interface to edit constraints and modify how they are evaluated in Constable. Users can change whole phrases, like “the latitude of the place” as well as individual elements like “the place”. The user can move the mouse over the English paraphrase of the constraint, and the system will automatically highlight meaningful phrases within the paraphrase that correspond to syntactically well-formed units of text. Once a phrase is selected to be changed, the lower window in the editor would then propose different phrases that could be used to change the phrase selected, grouped under headings like “information about the result” and “do something with the result”. A number of options will be shown, in our example those would include “add a number of hours to the result” and “check that the result is no later than a time point”. In this example, we would select the former and add one hour to the moonrise time returned by the information agent.

We also developed a graphical editor for constraints, shown in Figure 5. With this editor, users can view the different subprocedures used by Constable to evaluate a constraint, and can manipulate graphically the relationships among procedures and the information they use. This editor was prototyped and demonstrated in several scenarios and use cases in the Special Operations domain.

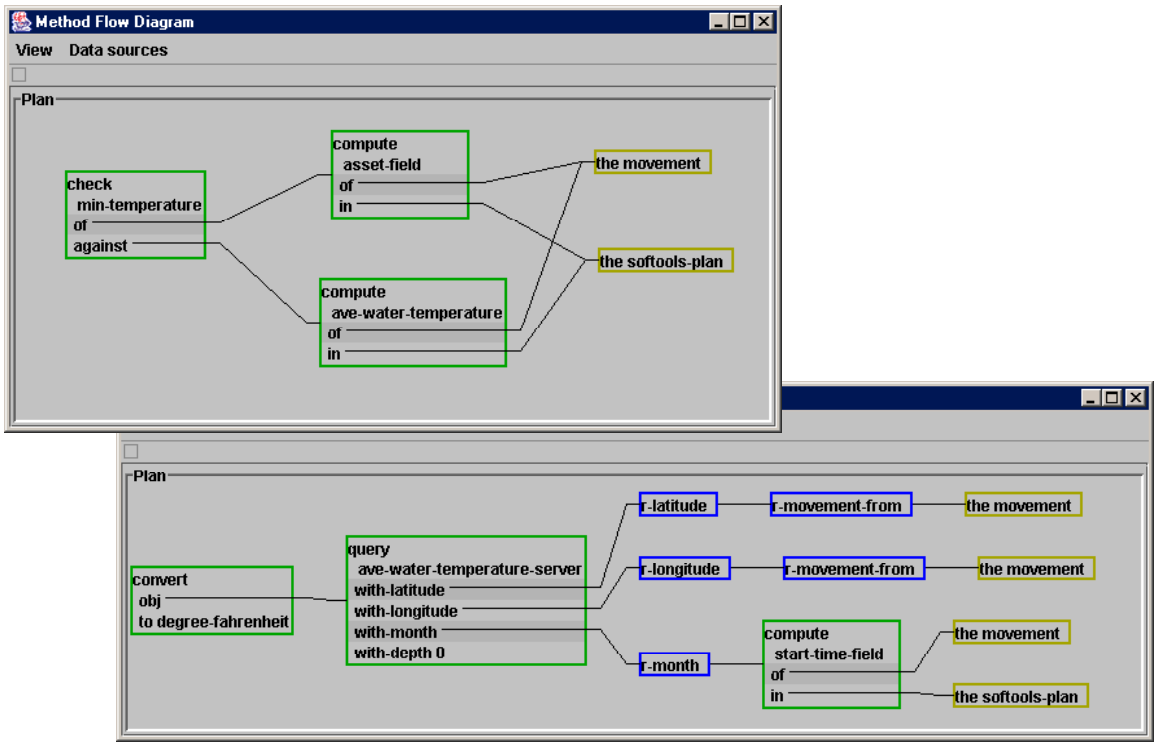

Figure 5. Modifying a constraint in Constable with a graphical editor.

2.2 A Planning Constraint Language for Active Templates

Several tools developed within the Active Templates program make use of explicit representations of constraints, and these have influenced the development of this representation language. For example, the Heracles project uses constraints to describe the data that is gathered from web wrappers, and constraints on combinations of values in fields. Its constraint representation is compatible with the one described here. CODA uses constraints to describe when a change in part of a plan should be communicated to a

separate planning group. In COMIREM, constraints are used for reasoning about resource allocation in plans. Tools for active forms, including those from SoftPro, BBN and Alphatech, can attach constraints to elements in forms to provide information about legal values and to compute values from other elements.

Based on discussions with these other groups, we created a language that can be used to communicate information about constraints in tools built under the Active Templates program. These tools refer to constraints in order to apply them to particular values, to advertise their input and output specifications to other tools, and to define how they are computed. The language allows tools to communicate with each other about constraints and also to store them in a central location in a tool-independent way.

We use a general definition for "constraints". Constraints are represented functionally and can be applied to a number of arguments to give a return value. Depending on how this value is interpreted, constraints can be used for a number of purposes, including the following:

- Computing the true or default value of an element in a template given values for other elements,
- Computing a range, or set of allowable values for an element in a template given values for other elements,
- Deciding whether a particular combination of values is allowable or in violation of the constraint,
- Deciding whether one value is preferred over another under the terms of the constraint.

In order to reason about constraints, some tools may also make use of further information such as a measure of the constraint's relative or absolute importance, ranges of values that are allowed, marginal or disallowed, and recommendations for recovering from constraint violations. The language provides a way to represent this information,

however, there is no requirement that other tools make use of this information in order to comply with the constraint specification language.

Figure 6 shows the rendering of one of Constable's constraints in this language. The constructs of the language are shown in XML.

A complete specification of this language was released to the Active Templates web site as a document titled "Constraints for Active Templates", authored by Jim Blythe.

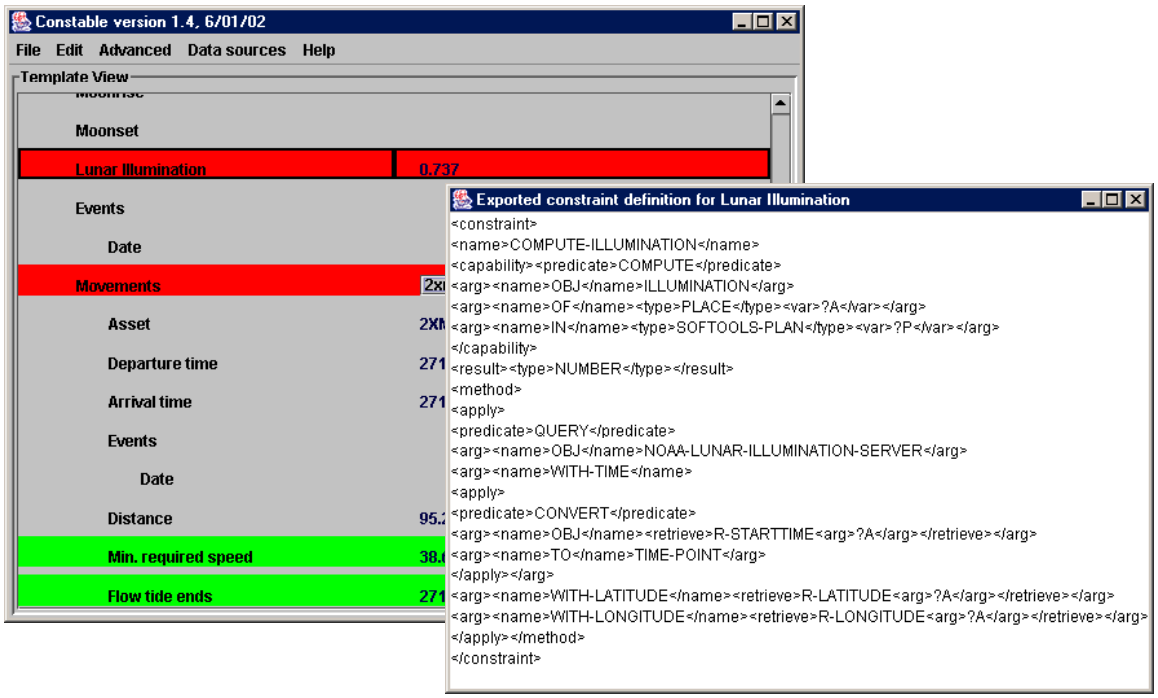

Figure 6. The language developed for exchanging constraints, showing a Constable constraint in the right-hand side rendered in XML.

2.3 Visualization as a Constraint Elicitation Technique

When human planners are confronted with many different choices that may generate a very large set of alternatives, it is hard for them to come up with a concise set of constraints that allows them to differenciate better from worse choices. We investigated the use of techniques to visualize the relative quality of various alternative plans and to see how they differ across significant features.

We developed a tool called VEIL (Visual Exploration and Incremental eLicitation) that integrates incremental utility elicitation with visual exploration of alternatives. VEIL provides a visualization of many alternatives that emphasizes some of their important features, and uses a technique called incremental utility elicitation to build a model of the user's preferences and make suggestions. VEIL provides an information-rich environment for the decision maker, while using an incremental utility estimate to help guide the search for a good alternative. The utility estimate can be updated based on preferences that are expressed directly in the exploration tool, or by modifying the weights.

This approach has three main features. First is the use of a visual exploration tool to display alternatives combined with incremental utility elicitation. The user can view alternatives through either custom or user-generated two-dimensional projections and can drill down on the information about a particular alternative. At the same time visual feedback is presented on the system's current utility estimates for all the displayed alternatives. The user can provide utility information either by changing weights or by indicating a preference between two alternatives directly in the interface.

Second, it uses a novel scheme for utility updates based on the set of preferences from the user. The system represents a utility function as a linear combination of constraint features. The update scheme maintains the utility weights as the vector solution of a linear programming problem whose constraints correspond to the user's preference vectors. This leads to an efficient update scheme that maintains an incremental utility estimate that is relatively stable, and provides an efficient test for violation of the linear utility assumption.

Third, the system uses a more expressive language than many systems performing incremental utility elicitation. The utility model is a linear combination of attributes that are themselves open to modification and creation by the user. For example, while preferring early start of operations, the user can easily incorporate a delay over the start time based on tides, essentially linking the two attributes. The system will also suggest modifications to the attributes when the assumption of a linearly additive utility is violated.

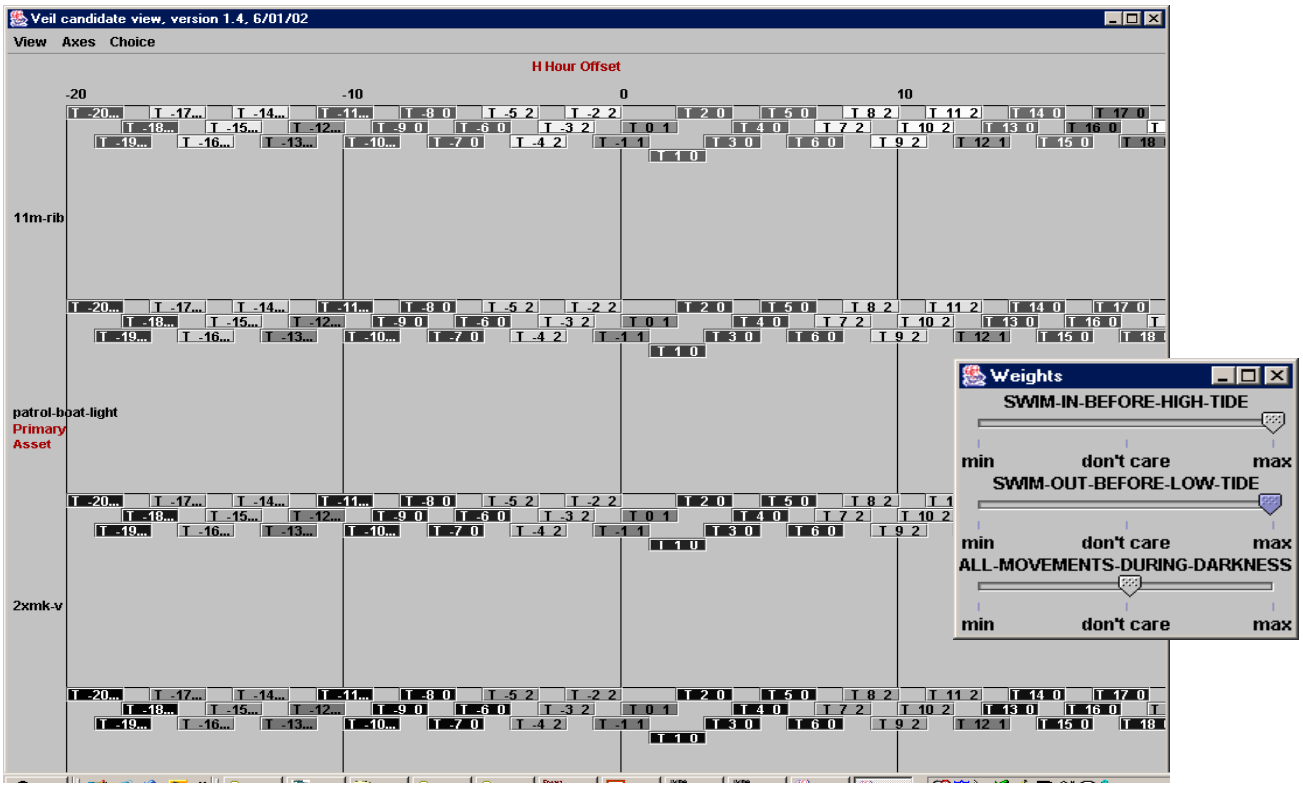

Figure 7. A snapshot of VEIL comparing different plans, the overlay shows how the user can manipulate the weights of the different factors evaluated.

Users interact with a two-dimensional projection of the alternatives corresponding to two attributes used for the X and Y coordinates, and optionally others depicted with shape, width, height and a textual label. This display broadly follows the starfield

approach, which is a well-known technique for displaying large sets of multi-attribute data. Two features of the layout are chosen to help the user find high-utility alternatives easily. First, the grey-scale shading of each rectangle is chosen to represent its rank in the ordering based on utility, with the best alternative in white and the worst in black. Second, when the rectangles for alternatives overlap, the alternative with higher estimated utility is placed on top.

Figure 7 shows a projection of alternatives for a beach landing operation with various alternative types of assets and the offset from the H-hour. The system begins with a default set of weights on the plan constraints. The user can alter the weights by hand. Optionally, while the user explores the alternatives, the plan browser updates its weights based on user actions. The system converges quickly, since the set of possible weights is much smaller than the space of alternative plans. Users can change the projection at any time by choosing the attributes to use for each axis, and the projection is automatically scaled. This layout was chosen because no rectangle is likely to completely obscure another. There may be many alternatives, which are summarized by representative ones that fit within the layout. Users can select a rectangle to bring it to the front of the display or to see more information about that choice in a separate window.

2.4 Integration with Special Operations Planning

Constable contains a set of default constraints on equipment, whether, and climatology from USSOCOM M525-6. This was done upon recommendation from the subject matter experts of the Active Templates program, so that Constrable could be demonstrated to Special Operations planners with constraints that were meaningful and useful in their planning tasks.

These constraints are tied to on-line data sources that Heracles/DataAgent (a tool developed by another group within the Active Templates program) provides. This allows users to define constraints that manipulate data coming from on-line sources. The integration required the following support for each data source: 1) encapsulate the query

to abstract away input details, 2) describe the expected types of the inputs in a schema, and 3) select relevant portions of the answer to a query. This information is kept up-to-date with each new release of Heracles/DataAgent.

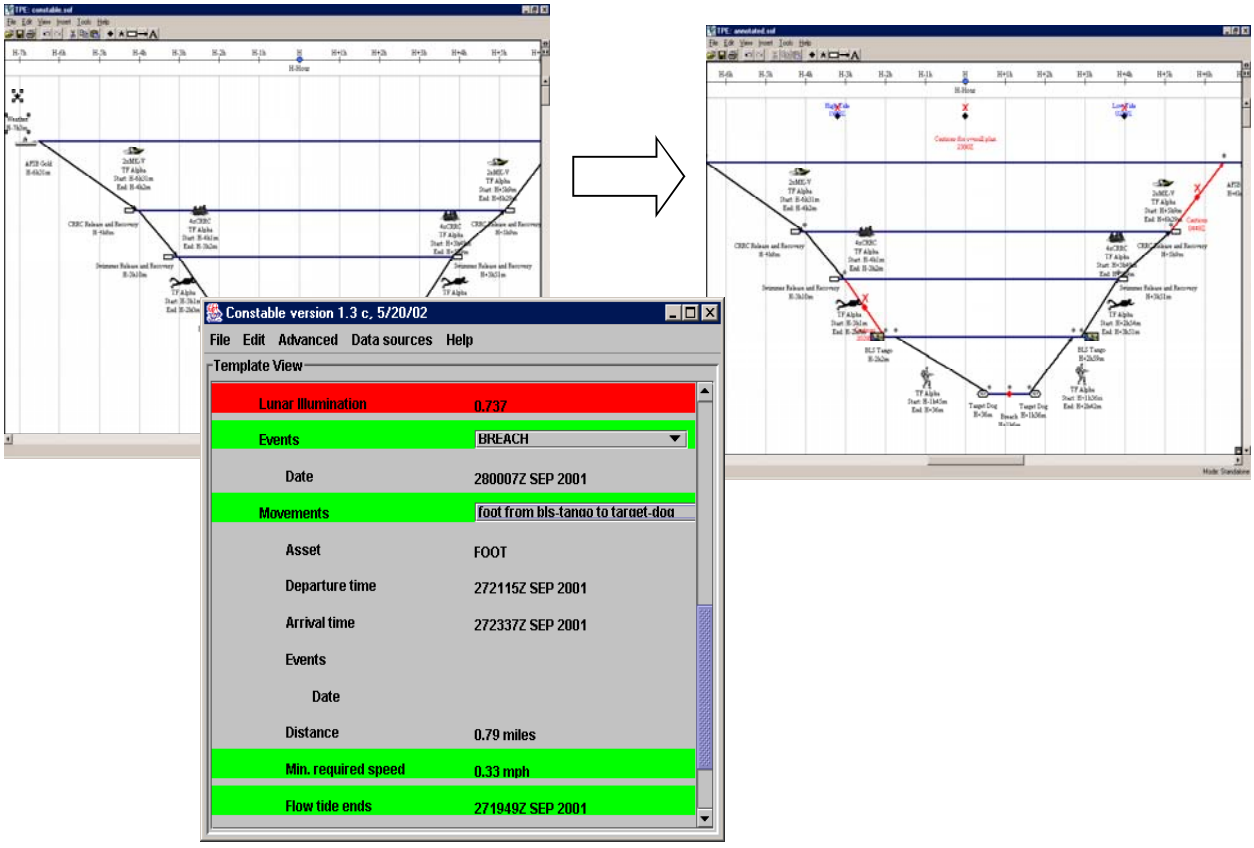

Figure 8. Integration of Constable with SofTools.

Constable can import a plan from SOFTools and evaluate it, putting the results back in SOFTools. This is shown in Figure 8.

3 Project Publications

“Constable User Guide”, Jim Blythe. Project report published on the Active Templates program web site. January 2003.

“Constraints for Active Templates”, Jim Blythe. Project report published on the Active Templates program web site. December 2001.

"Visual Exploration and Incremental Utility Elicitation", Jim Blythe. In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02).

"Integrating Expectations from Different Sources to Help End Users Acquire Procedural Knowledge", Jim Blythe. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01).

"An Integrated Environment for Knowledge Acquisition", Jim Blythe, Jihie Kim, Surya Ramachandran, Yolanda Gil. In Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI-2001), Santa Fe, New Mexico, January 2001. Recipient of the Best Paper Award.

Appendix I: Constable User Guide

Overview

Constable is a *Constraint Editing Tool* that helps users modify or add knowledge to an intelligent computer system. This version of Constable is able to critique plans created in Softools. This is done with a number of customizable checks on the plan called *constraints*. Examples of constraints are that an asset used in a movement should be either a CRRC or a Mk-V, or that the lunar illumination should be less than 0.5. In the following examples and activities, you will apply constraints to a sample Softools plan.

The following sections use exercises to illustrate some of the things you can do with Constable. They are:

- Introduction to the interface
- Modifying fields and their constraints
- Using the editor to change a constant value in a constraint
- Using the editor to change the way a field (or its constraint) is calculated
- Adding a new field to the plan view
- Using the editor to make a field (or constraint) value depend on another value

Introduction to the Interface

The first window that you will use in modifying or defining a constraint is the *Plan View*, which displays a number of different information fields for the plan. Two other windows will be used in this guide, the *Sources Window* and the *Method Editor*. The sources window shows all of the information that was used when the plan is checked against a particular constraint. The method editor can be used to modify constraints. It is described with examples in the last three sections of the guide.

The user sees the Plan View window when the application first starts. Figure 1 shows an example Plan View window. It displays information about the plan's fields and constraints. For example, the field labeled "`Lunar illumination`" has "`0.737`" as its value. The red background for the field and its value indicates that a constraint on this field was not satisfied.

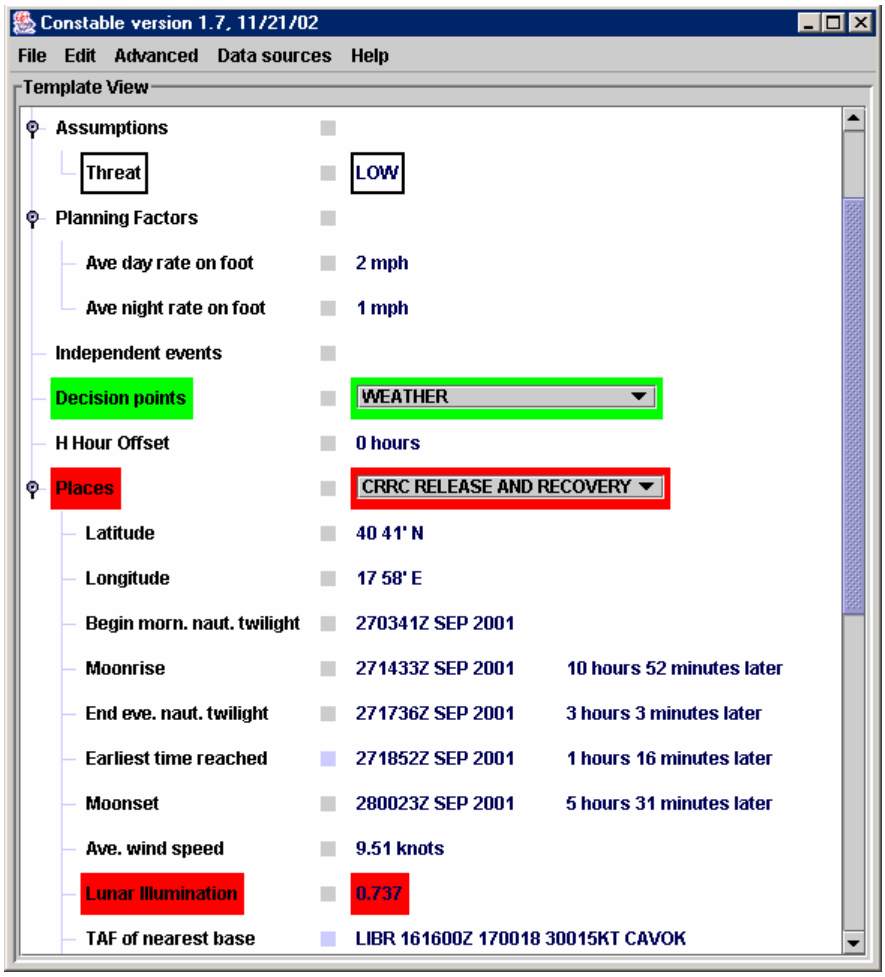

Figure 1. Plan Evaluator

In general, Constable can use constraints that check a field in many different ways, making use of background information about locations and equipment, and other planning factors and assumptions. Constable gives help for defining certain kinds of constraints, for example a constraint that involves a numerical value and checks against a minimum or maximum value. If the value of a field is not numerical, for instance "CRRC", the constraint can check if it is a member of a set of preferred values, or a set of values to be avoided. A field and its constraints can be edited using several tools from the menu that appears when you right-click over the field, as described in the next section.

Modifying fields and their constraints

Units may have different preferences about the lunar illumination, depending on the activities or enemy capabilities. In this section and the next one, we increase the maximum value in the constraint that Constable checks for the illumination from 0.3 to 0.8.

- Right-click on the **`Lunar Illumination`** field to bring up the edit menu.
- Select 'show sources' from the menu

A window is created with the title 'Summary of sources for Lunar Illumination', showing all the information that Constable used to compute and check the lunar illumination field. When field values are computed and constraints are checked, Constable may combine information from many sources, including other constraints that may solve a part of the problem. The first panel shows that constraints from two general groups were used. The second panel, labeled 'Live Data Sources', shows that some of the information came from querying external sources such as web sites. The third panel, labeled 'Information used', summarizes the data that was used in order to compute and check the field. In this case, the illumination is for the place called 'AFSB-GOLD' and its latitude, longitude and start-time (a Softools field) were used. The fourth panel, labeled 'Assumptions' shows some of the constraints that were used to check the field, including a 'maximum allowed value' which will be changed in this example.

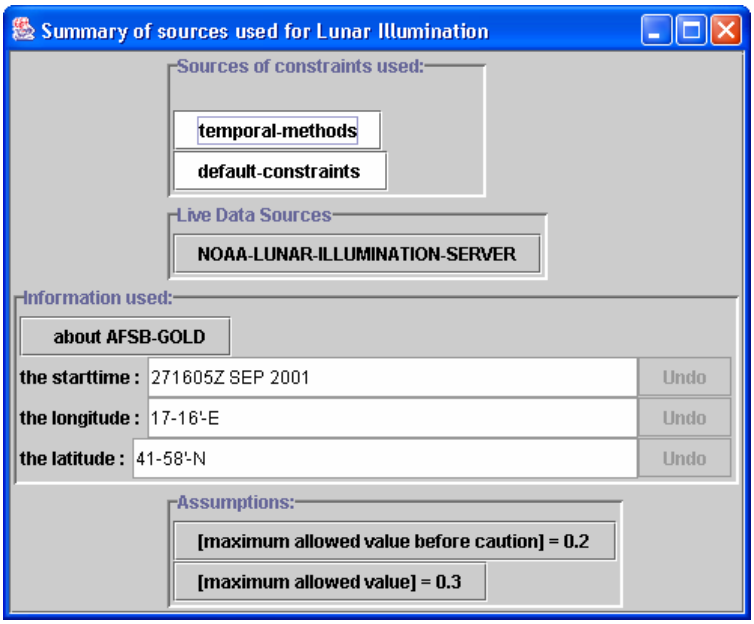

Figure 2. Information sources for the constraint "`Lunar Illumination`"

You can double-click on any of the pieces of information to change them using a specialized editor. You can also click on any of the lines in the Assumptions panel, like "maximum allowed value" to bring up an editor that can change how the assumption is made, as the next section shows.

Using the constraint editor to change a value

You can change the way any constraint works using the constraint editor. Following on from the previous section, we are editing the Lunar Illumination field and want to increase its maximum allowed value to 0.8:

- Click on the assumption that reads "[**maximum allowed value**] = **0.3**"

The editor should appear. This has three main sections (see Fig. 3). The first section at the top displays the current definition of the constraint. The first part of the definition begins with "`In order to…`" and is followed by a small description of what the constraint does. Here, it is followed by "`estimate a maximum allowed value of the Illumination for a place in a plan.`" The second part of the definition is found below. This shows how the constraint does its task; in this case, it always returns the number 0.3.

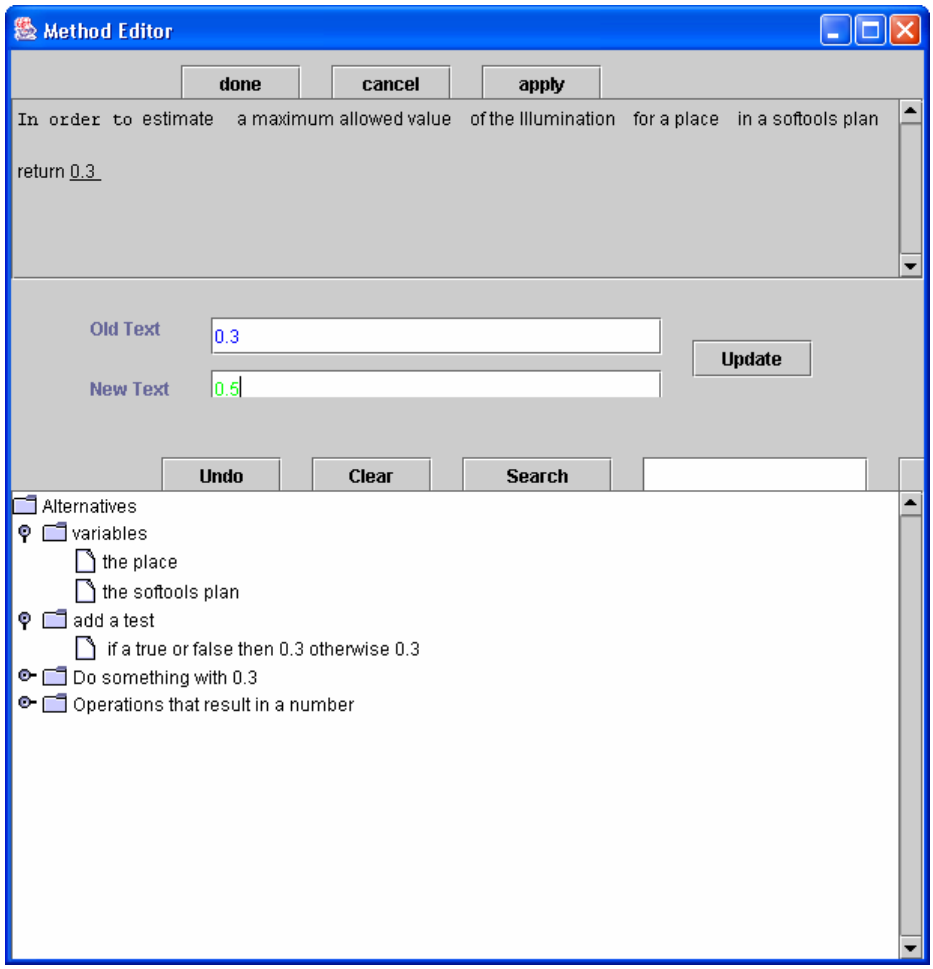

Figure 3. The Editor used to change the maximum allowed illumination to 0.5.

- Click over the **0.3** in the second part of the definition.

The number "**0.3**" will be underlined and displayed in blue in the middle section of the Editor, next to the label "Old text".

The bottom section of the Editor contains suggestions for how to change the selected text. These suggestions are organized into categories that group the related method activities. The methods for each constraint can have more complex definitions than simply having constant numbers like "0.3". For example, they can use simple programming languages that make them very powerful. The suggestions window in the Editor helps you piece together program statements. For instance, if you look under the "do something with 0.3" header you will find options such as "add 0.3 to a number" and "determine whether 0.3 is a member of a set". If you select one of these options, it appears in the "with" field in the middle of the Method Editor. Then you can use it to replace text in the method definition by clicking "**Update**".

- Type the replacement text, in this case "**0.8**", in the line next to the "**new text**" label below the window that displays "0.3".

- Click the **"Update"** button to make the change in the method description of the editor's upper window.

The "Undo" button undoes any change made in this way. The "Clear" button de-selects the text you had selected without making any change.

- Click on the "**done**" button at the top of the Method Editor window.

You have just modified the lunar illumination constraint with a new maximum value. The new definition will be applied, after a short pause, in the Plan View. Since the maximum illumination, 0.8, is more than the actual illumination, 0.737, the constraint is satisfied and the field's border will change from red to yellow. It is yellow rather than green because a separate constraint, 'maximum allowed value before caution' still has a value of 0.2, so a caution is raised. This can also be changed through the assumptions panel.

Use the editor to change how a field is calculated

The editor can be used to change the way that fields or constraints are computed, as well as to change constants as in the previous section. Suppose that, because of a range of mountains, moonrise is effectively an hour later than the value from astronomical calculations. Rather than give a fixed value, we need Constable to add an hour to any value that is computed.

- Right-click on the "**begin morn. naut. twilight**" field and choose "**show constraint details**" from the menu.
- Click on the button that reads "Compute a moonrise field of a place in a softools plan"

The “constraint details” window shows a short summary of how the field is computed and checked. The constraint definition shows how the field value is computed for a place in a Softools plan: first it computes the earliest time the place is reached, then it makes a query to a Fetch agent for the beginning of nautical twilight at this time and location. As before, you can click over an item in the window to change it, and the editor will suggest terms to change it to. For example, if you click on “the chosen date”, the editor will show other information about dates that could be used in the constraint. You can try this, then click “clear” to get back to the original screen.

You can change whole phrases, like “the latitude of the place” as well as individual elements like “the place”. To change a whole phrase, move the mouse to the left of the phrase until it shows completely in blue, then click the mouse left button. For example, you can select the whole phrase ‘query Noaa Nautical Twilight … for TwilightBegins’ by clicking just to the left of the word “query”, or on the word “and” to the left of that.

- Click over the word “and” in the method definition window

The options window, the lower window in the editor, now shows different phrases that you could change the phrase into, grouped under headings like “information about the result” and “do something with the result”.

- Double-click on “Do something with the result” to see its choices.

A number of options will be shown, including “add a number hours to the result” and “check that the result is no later than a time point”. Since we want to add one hour to the moonrise time returned by the wrapper, select the first option.

- click on “add a number hours to the result”, then click “update”.

The option first goes into the “new text” line in the editor, then is used in the method definition when you click “update”. You will see that the editor now shows the phrase “a number” in red, and in the lower window displays “’a number’ is a placeholder that should be filled in”. We will change the placeholder to the number “1”, the same way that the constant was changed in the previous section:

- Click on “a number” in the definition in the editor. The text appears in the “Old text” field.
- Type “1” in the “New text” field of the editor.
- Click “update”. The definition’s last line now reads “add 1 hours to the time point”.
- Click “done”.

Constable will display a new window showing all the work that will be done when the modified constraint is applied. This is useful to check that the modified constraint is still consistent with Constable’s other knowledge. You can ignore this window and remove it by clicking on the standard windows button to kill the window.

As before, the value will change in the Plan View window: twilight is shown as beginning one hour later, and the second number on the moonrise field below, showing the time elapsed after the nautical twilight begins, is reduced by one hour.

Use the editor to make a field value depend on another value

Suppose that the maximum illumination allowed is higher when there is high average wind speed, perhaps because spray will interfere with visibility. We can use the editor to make the constraint reflect this, by making the maximum value that is computed depend on the wind speed.

- Right-click the **`Lunar illumination`** field and select '**show sources**'.

- Click on the assumption, "`[maximum allowed value] = [0.8]`"

The description for this method shows that it always returns 0.8, which was set in an earlier example.

- Click over "0.8".

Among the options that appear in the lower window is the heading "`add a test`", with one sub-heading.

- Click on the line "**`If a true or false then 0.8 otherwise 0.8`**"; and then
- Click on "**`Update`**" to modify the method to use this line.

The method now makes an as yet unspecified test, denoted in the description as "`a true or false`", and produces the same number whether the test turns out to be true as shown after the word "`then`" or false as shown after the word "`otherwise`".

The phrase "`a true or false`" appears in red at the top window of the editor. This means that the method cannot be used with its current definition, and that the phrase "`a true or false`" will need to be changed before the method can be used. The lower window now contains the line "`Suggestions for what to do next:`" followed by the line "`'a true or false' is a placeholder that should be filled in`". This means we must replace this phrase with a statement that will actually produce either true or false when it is used in the method.

Next, modify both the test and the set returned if it is true by performing the following steps.

- Click on "**`a true or false`**" in the upper window.

- Type "**wind speed**" in the field next to the "**`search`**" button in the middle of the editor window.

- Click "**Search**".

  One suggested term is "`compute Ave wind speed of the place in the plan`".

  The search tool finds complete phrases that match the terms you type. Shorter terms like 'wind speed' or 'wind' will also include in this phrase.

- Click on "**compute the ave wind speed of the place in the plan**". You will see that a new set of alternatives appears in the lower window, indented below the line that was selected. These alternatives build on the destination – for example, one group is "`do something with the result`". Show this group by double-clicking the line if necessary.

- Below the sub-heading "**do something with the result**", select "**check that the result is greater than or equal to a number**".

- Click "**Update**" to enter this test.

  As shown in Figure 4, the method description will now read:

```
compute Ave Wind Speed of the place in the softools plan
and  if  check that the knots is greater than or equal to a number
then 0.8
otherwise 0.8
```

The phrase "`a number`" appears in red and a line in the lower editor window warns you that this is a placeholder, just as "`a true or false`" was before you made the last change.

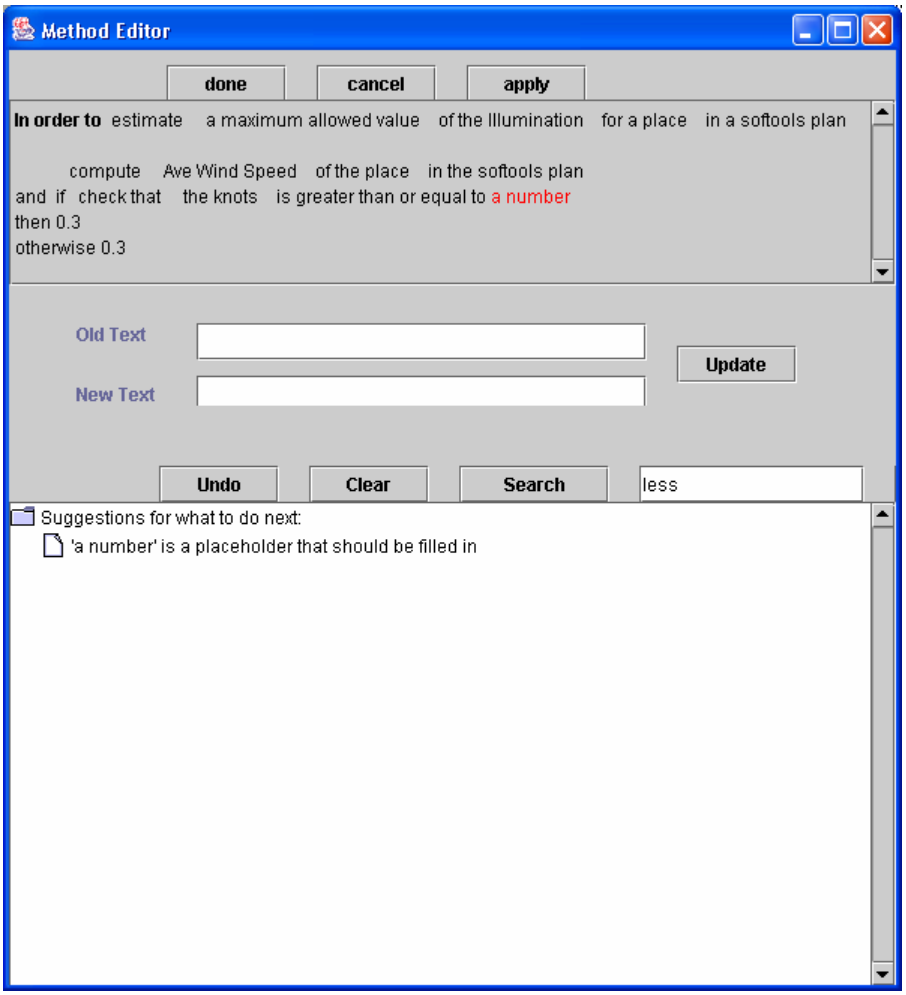

Figure 4: the editor after updating the condition

Next change "`a number`" 10:

- Click over "**`a number`**" in the definition in the upper window of the editor. Type "10" in the "new text" line and select "update".

Now select the number returned when the test is false:

- Click over the number "**`0.8`**" after "**`otherwise`**" in the second line of the method description.
- Type "0.3" in the "with" line of the editor.
- Click "update"
- Click "done".

When the constraint has been updated, the Lunar Illumination field will go back to red, because the wind is not strong enough to use the higher limit on illumination.